The Journal of Logic and Algebraic Programming 76 (2008) 226–245

Contents lists available at ScienceDirect

The Journal of Logic and Algebraic Programming

journal homepage: www.elsevier.com/locate/jlap

Soft Linear Set Theory

Richard McKinley [1]

Theoretische Informatik und Logik, Institut für Informatik und angewandte Mathematik, Neubrückstrasse 10, CH-3012 Bern, Switzerland

ARTICLE INFO

Available online 19 May 2008

ABSTRACT

A formulation of naïve set theory is given in Lafont's Soft Linear Logic, a logic with polynomial time cut-elimination. We demonstrate that the provably total functions of this set theory are precisely the PTIME functions. A novelty of this approach is the representation of the unary/binary natural numbers by two distinct sets (the safe naturals and the soft naturals).

1. Introduction

This paper concerns naïve set theory: systems of set theory where, given any predicate A, one may form the set $\{x \mid A\}$ of sets satisfying A. This rule is known as *unrestricted comprehension*. Russell's paradox famously demonstrates that this is inconsistent with classical logic. The observation that contraction is essential for Russell's paradox, and that moreover the system given by adding unrestricted comprehension to what is now known as **MALL** is consistent, seems to have been made first by Grishin, in [9] (see [10] for an exposition in English of these results). By contrast, unrestricted comprehension is incompatible with the exponentials of Linear Logic; one can easily reconstruct Russell's argument in this context. Finding systems of intermediate expressivity between **MALL** and **LL** compatible with unrestricted comprehension hinges on the *complexity* of cut-elimination.

Girard, in his paper Light Linear Logic [7], introduces the notion of intrinsic polytime normalization, whereby a logical system (a system of sequents, proof nets or lambda terms) has normalization polynomially bounded by some property of the proofs/terms, independent of the complexity of any cuts involved. Thus, for example, a proof net in Light Linear Logic normalizes after a number of steps bounded by a polynomial whose degree depends only on the nesting of its exponentials. Girard makes the observation that it is precisely this property (bounds on cut-elimination independent of cut-rank) which allows for a consistent extension into naïve set theory, and gives a sketch of a set theory based on Light Linear Logic in the appendix of [7], including an (unproved) claim that the provably total functions of this system are precisely the polytime functions.

Owing to complications in the proof theory of Light Linear Logic, details of a set theory with light exponentials did not appear until Terui [14] established the polytime representation property for Light Affine Set Theory (**LAST**). **LAST** is based on Light Affine Logic [1], a system which, by virtue of its unrestricted weakening, has a simpler presentation as a sequent calculus.

While light logics have been very successful in capturing the polytime functions, they suffer from the presence of the paragraph modality §, meaning that light logics are not subsystems of Linear Logic. Lafont's Soft Linear Logic [15] is another logic which captures the polynomial time functions. Unlike Light Linear/Affine Logic, it is a fragment of linear logic (that is, it does not include the paragraph modality), and additionally it has a very simple sequent calculus presentation. It is natural to

E-mail address: mckinley@iam.unibe.ch
URL: http://www.iam.unibe.ch/mckinley
[1] Work supported by the Swiss National Science Foundation Grant "Algebraic and Logical Aspects of Knowledge Processing."

1567-8326/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jlap.2008.02.010

consider whether SLL with unrestricted comprehension also captures the polytime functions. This is the question addressed in this paper. We will see that this is the case, in the following sense: we give a notion of provably total function for the system such that the provably total functions are precisely the polytime computable functions. The presence of unrestricted comprehension gives us access to fixpoints of arbitrary formulae, which we use to give a common codomain to each polytime function on $\mathbb{N}$: thus the work is an improvement both on the **LAST** approach to set theory (where the definition of totality contains a variable number of occurences of §) and on the representation of functions in **SLL** (where it is not know how to give a common codomain for each representable function).

2. Soft Linear Logic

Soft Linear Logic [15] is a system based on the same language as Linear Logic [5], and whose cut-elimination enjoys a polynomial bound. The logic arises by observing that the usual exponential rules of (intuitionistic) linear logic

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A} \qquad \frac{\Gamma, A \vdash C}{\Gamma, !A \vdash C} \qquad \frac{\Gamma, !A, !A \vdash C}{\Gamma, !A \vdash C} \qquad \frac{\Gamma \vdash C}{\Gamma, !A \vdash C}$$

are interderivable with the rules *soft promotion, digging and multiplexing*:

$$\frac{\Gamma \vdash A}{!\Gamma \vdash !A} \qquad \frac{\Gamma, !!A \vdash C}{\Gamma, !A \vdash C} \qquad \frac{\Gamma, A^{(n)} \vdash C}{\Gamma, !A \vdash C}$$

Second-order Soft Linear Logic ($\mathbf{SLL}_2$) is the fragment of second-order (intuitionistic) Linear Logic with the usual exponentials replaced by soft promotion and multiplexing (there is also a classical version of **SLL**). Note that since the system lacks digging, it cannot prove the usual !-contraction rule of Linear Logic.

Lafont gives a system of proof nets for this logic, and demonstrates that each net reduces to a unique normal form in a polynomially bounded number of steps— this bound has degree given by the nesting of exponentials in the proof net.

Lafont proceeds to define a type of natural numbers

$$N := \forall\alpha.!(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha$$

and to give System F style representations of functions on those natural numbers. A quirk of the system is that these functions are not typable $N \multimap N$, or even $!N \multimap N$; for example, successor is represented by the following proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\alpha \vdash \alpha \quad \alpha \vdash \alpha}{\alpha, \alpha \multimap \alpha \vdash \alpha}\multimap L \qquad \alpha \vdash \alpha}{\alpha, \alpha \multimap \alpha, \alpha \multimap \alpha \vdash \alpha}\multimap L}{\alpha \multimap \alpha, \alpha \multimap \alpha \vdash \alpha \multimap \alpha}\multimap R \qquad !(\alpha \multimap \alpha) \vdash !(\alpha \multimap \alpha)}{!(\alpha \multimap \alpha), (\alpha \multimap \alpha), !(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha \vdash \alpha \multimap \alpha}\multimap L}{(!(\alpha \multimap \alpha) \otimes (\alpha \multimap \alpha)), !(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha \vdash \alpha \multimap \alpha}\otimes L}{!(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha \vdash (!(\alpha \multimap \alpha) \otimes (\alpha \multimap \alpha)) \multimap \alpha \multimap \alpha}\multimap R}{\forall\alpha.!(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha \vdash \forall\alpha.(!(\alpha \multimap \alpha) \otimes (\alpha \multimap \alpha)) \multimap \alpha \multimap \alpha}\forall L, \forall R$$

where the type on the right-hand side differs from that on the left-hand side. In general typing varies from function to function.

Lafont gives a type B of booleans, and demonstrates that for any polytime decidable predicate $A(w)$ on the boolean words W, there is a $\mathbf{SLL}_2$ proof of $W^n \vdash B$ corresponding to that predicate; this completes the proof that $\mathbf{SLL}_2$ captures polytime.

3. Soft Linear Set Theory

3.1. Syntax

Our syntax mirrors that of [7,14], the major difference being the lack of a paragraph modality:

Definition 1 (*Soft Linear Set Theory* ***SLST***). The *terms* and *formulae* of **SLST** are defined simultaneously as follows:
- Term variables $x,y,z,\ldots$ are terms;
- If A is a formula and x is a term variable then $\{x \mid A\}$ is a term;
- If t and u are terms then $t \in u$ is a formula;
- **0** and **1** are formulae;
- If A and B are formulae then the following are formulae: $A \otimes B$, $A \multimap B$, $A \oplus B$, $!A$;
- If A is a formula and x is a term variable, then $\forall x.A$ and $\exists x.A$ are formulae.

We use $t,u,v,\ldots$ to denote sets, $A,B,C,\ldots$ to denote formulae, and $\Gamma,\Delta,\Sigma,\ldots$ to denote multisets of formulae. If Γ stands for $A_1,\ldots,A_n$, then $!\Gamma$ stands for $!A_1,\ldots,!A_n$. The notation $A^{(d)}$ stands for $\underbrace{A,\ldots,A}_{d\text{times}}$, the notation A^d for $\underbrace{A \otimes \cdots \otimes A}_{d\text{ times}}$, and the notation $!^dA$ for $\underbrace{!\ldots!}_{d\text{ times}}A$. We use $A \multimap\!\!\circ\, B$ as shorthand for $(A \multimap B) \otimes (B \multimap A)$.

We will also require linear versions of relativised quantifiers

$$\forall x \in s.A := \forall x.(x \in s \multimap A(x)) \quad \exists x \in s.A := \exists x.(x \in s \otimes A(x)).$$

The variable x is *bound* in $\{x \mid A\}$, $\forall x.A$ and $\exists x.A$. We will consider two terms which differ up to renaming of bound variables to be identical. We use the notation $u[x := t]$ to denote the term obtained from u by substituting t for all free occurrences of x. A similar notation is used for substitution into formulae.

The rules of **SLST** are given in Table 1, where contexts Γ,Γ' are multisets (and so exchange is implicit). Note that some of our rules here are redundant, in the sense that, given the rules for $\forall$, $\in$ and $\multimap$, and an arbitrary closed term t_0, the definition

$$\exists y.A := \forall x.(\forall y.(A \multimap t_0 \in x) \multimap t_0 \in x)$$

satisfies all the desired properties of our existential quantifier (here we simulate the properties of a second-order $\forall$, using comprehension to exchange a formula depending on formulae for a formulae depending on sets). However, since we are working in the absence of weakening the connective $\oplus$ is not derivable. Note also that we could just as easily give a classical version of **SLST**—since our goal here is to prove polynomial soundness and completeness it suffices to consider the intuitionistic fragment.

Theorem 2 (Cut elimination). *If A is provable in* ***SLST*** *, it is provable without using cut.*

Proof. By Girard's observations about unrestricted comprehension—since cut-elimination in **SLL** does not proceed by cut-rank, the extension of **SLL** by comprehension retains cut-elimination. □

Corollary 3. ***SLST*** *is consistent.*

Corollary 4
(a) *Disjunction property: If $\vdash A \oplus B$ is provable, either $\vdash A$ or $\vdash B$ is provable.*
(b) *Existence Property: If $\vdash \exists x.A$ is provable, then $\vdash A[x := t]$ is provable for some term t.*

3.2. General substructural set theory

Before approaching the behaviour of the soft modality in set theory, we recall some standard properties of naïve set theory in the absence of contraction (and weakening). For more details see [12,14].

We may define an equality on terms of **SLST** by the identity of indiscernibles (Leibniz's law)—that is, two individuals are equal if they have identical properties (where here the notion of property is given by set membership).

Definition 5 (*Leibniz Equality*).

$$t = u := \forall x.(t \in x \multimap u \in x).$$

The following are easy to verify:

Proposition 6
- $\vdash t = t$
- $\vdash t = u \multimap (A[x := t] \multimap A[x := u])$

Table 1
Soft Linear Set Theory

$$\frac{}{A \vdash A}\,\mathrm{Ax}$$

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\otimes\mathrm{L} \qquad \frac{\Gamma \vdash A \quad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \otimes B}\otimes\mathrm{R}$$

$$\frac{\Gamma \vdash C}{\Gamma, \mathbf{1} \vdash C}\,\mathbf{1}L \qquad \frac{}{\vdash \mathbf{1}}\,\mathbf{1}R$$

$$\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C}\multimap\mathrm{L} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\multimap\mathrm{R}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\oplus\mathrm{R} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\oplus\mathrm{R} \qquad \frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\oplus\mathrm{L}$$

$$\frac{}{\Gamma, 0 \vdash A}\,\mathbf{0}\mathrm{L}$$

$$\frac{\Gamma \vdash C}{!\Gamma \vdash !C}\,\mathrm{SP} \qquad \frac{\Gamma, A^{(n)} \vdash C}{\Gamma, !A \vdash C}\,\mathrm{mplx}$$

$$\frac{A[x := t], \Gamma \vdash C}{\forall x.A, \Gamma \vdash C}\forall\mathrm{L} \qquad \frac{\Gamma \vdash C}{\Gamma \vdash \forall x.C}\forall\mathrm{R}$$

$$\frac{\Gamma, A \vdash C}{\Gamma, \exists x.A \vdash C}\exists\mathrm{L} \qquad \frac{\Gamma \vdash C[x := t]}{\Gamma \vdash \exists x.C}\exists\mathrm{R}$$

$$\frac{A[x := t], \Gamma \vdash C}{t \in \{x \mid A\}, \Gamma \vdash C}\in\mathrm{L} \qquad \frac{\Gamma \vdash C[x := t]}{\Gamma \vdash t \in \{x \mid C\}}\in\mathrm{R}$$

$$\frac{\Gamma \vdash A \quad \Gamma', A \vdash C}{\Gamma, \Gamma' \vdash C}\,\mathrm{Cut}$$

- $\vdash t = u \multimap u = t$
- $\vdash t = u \otimes u = r \multimap t = r$
- $\vdash t = u \multimap t = u \otimes t = u$
- $\vdash t = u \multimap \mathbf{1}$

We may now define some standard set theoretic operations:

Definition 7

$$\emptyset := \{x \mid \mathbf{0}\}; \qquad \{t\} := \{x \mid x = t\};$$
$$\{t,u\} := \{x \mid x = t \oplus x = u\}; \qquad \{t_1, \ldots, t_n\} := \{x \mid x = t_1 \oplus \cdots \oplus x = t_n\};$$
$$t \cup u := \{x \mid x \in t \oplus x \in u\}; \qquad \langle t,u\rangle := \{\{t\},\{t,u\}\};$$
$$\langle t_1, \ldots, t_n\rangle := \langle t_1, \langle t_2, \langle t_3, \ldots, \langle t_{n-1}, t_n\rangle \ldots\rangle\rangle.$$

Proposition 8. *The following are provable in **SLST**:*

- $\vdash t \notin \emptyset$;
- $\vdash t \in \{u\} \circ\!\!-\!\!\circ t = u$;

- $\vdash t \in \{t,u\} \circ\!\!-\!\!\circ t = u \oplus t = v$;
- $\vdash \langle t,u\rangle = \langle r,s\rangle \circ\!\!-\!\!\circ t = r \otimes u = s$.

We will find the following linear version of a standard abbreviation useful:

$$\exists^! x.A := \exists x.(A \otimes \forall y.(A[x := y] \multimap x = y))^2$$

In fact, Leibniz equality is nothing more than the internal representation of syntactic identity (i.e. alpha equivalence):

Proposition 9. $\vdash t = u$ *in* ***SLST*** *iff* t *and* u *are syntactically identical.*

Proof. **SLST** proves $\forall x.(t \in x \multimap u \in x), t \in x \vdash u \in x$. Hence, if $t = x$ is provable, $t \in x \vdash u \in x$ is also provable. By cut-elimination, this last sequent should be an axiom. Hence t and u are syntactically identical. The other direction is immediate. □

Strikingly, the axiom of extensionality

$$\forall x.(x \in t \circ\!\!-\!\!\circ x \in u) \circ\!\!-\!\!\circ t = u$$

is inconsistent with **SLST** (and naïve set theory in general), since from it we may derive unrestricted contraction (see [4,12]).

Naïve set theory also admits a powerful fixpoint theorem, which we will use heavily in this paper:

Theorem 10 (Fixpoint theorem [7,12,4]). *For any formula* A, *there exists a term* f *such that*

$$t \in f \circ\!\!-\!\!\circ A[y := f, x := t]$$

is provable for any t.

The fixpoint is given by the following: first define

$$s := \{\, z \mid \exists u.\exists v.(z = \langle u,v\rangle \otimes A[y := \{\, w \mid \langle w,v\rangle \in v \,\}, x := u]) \,\},$$

and then let the term f (the desired fixpoint of A) be

$$f := \{\, w \mid \langle w,s\rangle \in s \,\}.$$

The required properties may now be easily inferred.

4. Representing sets and functions in SLST

The terms of **SLST** should be regarded as (abstract, intensional) representations of sets. We make this notion formal:

Definition 11. A set S is *represented* by a term s of **SLST** if there is a bijection $(.)^*$ from S to the terms u such that $\vdash u \in s$ is provable in **SLST**.

Our goal is to show that the functions representable as terms of **SLST** are precisely the polytime functions. We give here two notions of the representation of functions in **SLST**. Both identify a function with a representation of its graph as a term of **SLST**, but they differ on the statement of totality. In what follows, let $\vec{T}$ denote $T_1 \times \cdots \times T_n$, $\vec{m}$ denote $(m_1, \ldots, m_n)$, where m_i is a member of the set T_i, and let $\vec{t_i}$ denote the (internal) **SLST** tuple $\langle t_1, \ldots, t_n\rangle$.

The first notion of representation is close in spirit to the encoding of function spaces in usual linear logic:

Definition 12. A function $\phi : T_1 \times \cdots \times T_k \to S$ is *represented* by a term f of **SLST** with domains $t_1, \ldots, t_k$ and codomain s if

(a) Each T_i and S are represented by t_i and s, respectively;
(b) $\vdash \langle \vec{m}^*, n^* \rangle \in f$ for any $\vec{m} \in \vec{T}$ and $n \in S$ such that $\phi(\vec{m}) = n$;
(c) $\vdash \forall x_1 \ldots \forall x_k.((!(x_1 \in t_1) \otimes \cdots \otimes !(x_n \in t_n)) \multimap \exists^! y(y \in s \otimes \langle \vec{x}, y\rangle \in f))$.

This definition is unsurprising in the context of linear logic. However, in **SLL** we cannot in general compose functions with type $!A \multimap B$ and another with type $!B \multimap C$. To compose these types in *LL* requires digging:

$$!A \xrightarrow{\text{digging}} !!A \xrightarrow{!f} !B \xrightarrow{g} C$$

Similar problems arise in the composition of representable functions. To allow, in certain special cases, composition of functions, we use the following notion of representability:

[2] Here we use the standard abbreviation for unique existence, despite the usage of an exclamation elsewhere for the soft exponential; without some abbreviation some already long expressions would become unreadable.

Definition 13. A function $\phi : T_1 \times \cdots \times T_k \to S$ is 0-represented by a term f of **SLST** with domains $t_1, \ldots, t_k$ and codomain s if

(a) Each T_i and S are represented by t_i and s, respectively;
(b) $\vdash \langle \vec{m}^*, n^* \rangle \in f$ for any $\vec{m} \in \vec{T}$ and $n \in S$ such that $\phi(\vec{m}) = n$;
(c) There exists natural numbers $n_1, \ldots, n_k$ such that

$$\vdash \forall x_1 \ldots \forall x_k.(((x_1 \in t_1)^{n_1} \otimes \cdots \otimes (x_k \in t_k)^{n_k}) \multimap \exists^! y.(y \in s \otimes \langle \vec{x}, y \rangle \in f))$$

is provable in **SLST**.

Clearly, 0-representability implies representability (via applications of multiplexing). We will write

$$f : t_1^{(n_1)} \times \cdots \times t_k^{(n_k)} \to s$$

if f is a term with the third property above. We refer to the number n_i as the *multiplicity* of t_i in f.

5. Tally integers

5.1. Soft and safe naturals

We now begin defining the sets used to represent Turing machines as terms of **SLST**, beginning with unary or tally numbers. The polytime functions are defined in terms of binary (or n-ary) numbers, and we will see in the next section the definitions of terms representing words over a finite alphabet. Time, on the other hand, is represented as a unary number. While we could use induction over the length of binary words to achieve the same effect, the example of unary numbers neatly illustrates some of the properties of **SLST**.

Following [14], we represent natural numbers via ordered pairs:

Definition 5.1

$$\mathsf{0} := \emptyset; \quad \mathsf{S}t := \langle \emptyset, t \rangle; \quad \mathsf{n} := \mathsf{S}^n\mathsf{0}.$$

Proposition 14
(a) $\vdash \mathsf{S}(t) \neq \mathsf{0}$.
(b) $\vdash \mathsf{S}(t) = \mathsf{S}(s) \multimap\!\!\circ\!\!\multimap t = s$.

We may now internally define the natural numbers in **SLST**, based upon the type of natural numbers in linear logic:

Definition 15 (*Soft natural numbers*).

$$\mathsf{N} := \{ x \mid \forall \alpha(!\forall y(y \in \alpha \multimap \mathsf{S}y \in \alpha) \multimap (\mathsf{0} \in \alpha \multimap x \in \alpha)) \}$$

Proposition 16. *The term* N *represents* $\mathbb{N}$ *in* ***SLST****. That is,* $\vdash t \in \mathsf{N}$ *iff* $\vdash t = \mathsf{n}$ *for some* $n \in \mathbb{N}$.

Thus, if a term t is provably in N, and for some other term s, we have $\vdash \mathsf{0} \in s$ and $y \in s \vdash \mathsf{S}y \in s$, by cut we have $\vdash t \in s$.
By instantiating α with $\{ x \mid \mathbf{1} \}$, we may derive weakening for soft naturals:

Proposition 17. *The following is provable in* ***SLST****:* $x \in \mathsf{N} \vdash \mathbf{1}$.

Proof. By the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{1} \vdash \mathbf{1}}{y \in \{x \mid \mathbf{1}\} \vdash \mathsf{S}y \in \{x \mid \mathbf{1}\}}}{\vdash y \in \{x \mid \mathbf{1}\} \multimap \mathsf{S}y \in \{x \mid \mathbf{1}\}}}{\vdash \forall y.(y \in \{x \mid \mathbf{1}\} \multimap \mathsf{S}y \in \{x \mid \mathbf{1}\})}}{\vdash !\forall y.(y \in \{x \mid \mathbf{1}\} \multimap \mathsf{S}y \in \{x \mid \mathbf{1}\})} \quad \dfrac{\vdash \mathbf{1}}{\mathsf{0} \in \{x \mid \mathbf{1}\}} \quad \dfrac{\mathbf{1} \vdash \mathbf{1}}{x \in \{x \mid \mathbf{1}\} \vdash \mathbf{1}}}{\mathsf{0} \in \{x \mid \mathbf{1}\} \multimap x \in \{x \mid \mathbf{1}\} \vdash \mathbf{1}}}{!\forall y.(y \in \{x \mid \mathbf{1}\} \multimap \mathsf{S}y \in \{x \mid \mathbf{1}\}) \multimap (\mathsf{0} \in \{x \mid \mathbf{1}\} \multimap x \in \{x \mid \mathbf{1}\}) \vdash \mathbf{1}}}{\forall \alpha.(!\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha) \multimap (\mathsf{0} \in \alpha \multimap x \in \alpha)) \vdash \mathbf{1}} \qquad \square$$

The soft natural numbers exhibit a form of induction, which we will call *Soft induction over* N.

Proposition 18. *The following inference is derivable in* ***SLST****:*

$$\frac{\Gamma \vdash A[x := 0] \qquad \Delta, A[x := y] \vdash A[x := \mathsf{S}y]}{\Gamma, !\Delta, t \in \mathsf{N} \vdash A[x := t]}\ \mathsf{N} - \mathit{ind}.$$

Proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Delta, A[x := y] \vdash A[x := \mathsf{S}y]}{\Delta, y \in \{x \mid A\} \vdash \mathsf{S}y \in \{x \mid A\}}\in \mathrm{L}, \in \mathrm{R}}{\Delta \vdash y \in \{x \mid A\} \multimap \mathsf{S}y \in \{x \mid A\}}\multimap \mathrm{R}}{\Delta \vdash \forall y.(y \in \{x \mid A\} \multimap \mathsf{S}y \in \{x \mid A\})}\forall \mathrm{R}}{!\Delta \vdash !\forall y.(y \in \{x \mid A\} \multimap \mathsf{S}y \in \{x \mid A\})}\mathrm{SP} \qquad \dfrac{\dfrac{\Gamma \vdash A[x := 0]}{\Gamma \vdash 0 \in \{x \mid A\}}\in \mathrm{R} \qquad \dfrac{A[x := t] \vdash A[x := t]}{t \in \{x \mid A\} \vdash A[x := t]}\in \mathrm{L}}{\Gamma, 0 \in \{x \mid A\} \multimap t \in \{x \mid A\} \vdash A[x := t]}\multimap \mathrm{L}}{\Gamma, !\Delta, !\forall y.(y \in \{x \mid A\} \multimap \mathsf{S}y \in \{x \mid A\}) \multimap (0 \in \{x \mid A\} \multimap t \in \{x \mid A\}) \vdash A[x := t]}\multimap \mathrm{L}$$

$$\frac{\Gamma, !\Delta, !\forall y.(y \in \{x \mid A\} \multimap \mathsf{S}y \in \{x \mid A\}) \multimap (0 \in \{x \mid A\} \multimap t \in \{x \mid A\}) \vdash A[x := t]}{\Gamma, !\Delta, t \in \mathsf{N} \vdash A[x := t]}\forall \mathrm{L}. \qquad \square$$

However, it does not seem possible to find a non-trivial predicate A such that $\exists y \in \mathsf{N}.A(x,y) \vdash \exists y \in \mathsf{N}.A(\mathsf{S}x,y)$ holds; there is no obvious proof even for $x = \mathsf{S}y$. Thus it seems at first that we cannot use this induction principle to reason about natural numbers in our system. Consider, however, the following set defined by a fixpoint (which exists by Theorem 10):

Definition 19 (*Safe natural numbers*).

$$x \in \mathsf{N}' \circ\!\!-\!\!\circ x = 0 \oplus \exists y(y \in \mathsf{N}' \otimes x = \mathsf{S}y)$$

This set also represents the natural numbers in **SLST**, but unlike N it is provably closed under successor.

Proposition 20. *The following are provable in* ***SLST*** :
(a) $\vdash 0 \in \mathsf{N}'$;
(b) $\vdash t \in \mathsf{N}' \vdash \mathsf{S}t \in \mathsf{N}'$;
(c) $\vdash t \in \mathsf{N}'$ *iff* $\vdash t = \mathsf{n}$ *for some* $n \in \mathbb{N}$.

Proof. The statements (a) and (b) are clear, and together they demonstrate one direction of (c). For the other, we reason by induction on the size of t.

Suppose $\vdash t \in \mathsf{N}'$. By the disjunction property either $\vdash t = 0$ or $\vdash \exists y \in \mathsf{N}'.(t = \mathsf{S}y)$ is provable. In the first case, t is syntactically identical to 0. Otherwise, by the existence property there is some u such that $\vdash u \in \mathsf{N}'$ and $\vdash t = \mathsf{S}u$. Thus t is syntactically identical to $\mathsf{S}u$, and thus u has smaller size than t. By the induction hypothesis, u is syntactically identical to m for some $m \in \mathbb{N}$, and thus t is syntactically identical to p, where $p = m + 1$. $\square$

Corollary 21. $\vdash t \in \mathsf{N}$ *if and only if* $\vdash t \in N'$.

Of course, this is a metatheorem, but we may derive one direction of the transformation via soft induction. In fact, we can do better.

Theorem 22 (*Soft coercion*). *For each natural number* n,

$$x \in \mathsf{N} \vdash !^n x \in \mathsf{N}'.$$

Proof. Fix an $n \in \mathbb{N}$. Then $\vdash !^n 0 \in \mathsf{N}'$ is provable in **SLST**, and $!^n t \in \mathsf{N}' \vdash !^n \mathsf{S}t \in \mathsf{N}$, from Proposition 20 and soft promotion. The result follows by soft induction. $\square$

Similarly, we obtain a form of contraction for safe naturals.

Theorem 23. *The following inference is derivable in* ***SLST****:*

$$\frac{t \in \mathsf{N}', t \in \mathsf{N}', \Gamma \vdash \Delta}{t \in \mathsf{N}, \Gamma \vdash \Delta}\ \mathsf{N}' - \mathit{cont}$$

Proof. We have $\vdash 0 \in \mathsf{N}' \otimes 0 \in \mathsf{N}'$ and $x \in \mathsf{N}' \otimes x \in \mathsf{N}' \vdash \mathsf{S}x \in \mathsf{N}' \otimes \mathsf{S}x \in \mathsf{N}'$. By soft induction, $t \in \mathsf{N} \vdash t \in \mathsf{N}' \otimes t \in \mathsf{N}'$. An application of cut completes the proof. □

We will need both N and N′ to build a arithmetic in **SLST**. Define the graphs of addition and multiplication as follows:

Definition 24. Let add be a term which satisfies

$$\langle x,y,z\rangle \in \mathsf{add} \multimap (y = 0 \otimes x = z) \oplus \exists y'.\exists z'.(y = \mathsf{S}(y') \otimes z = \mathsf{S}(z') \otimes \langle x,y',z'\rangle \in \mathsf{add}).$$

Such a term exists by the fixpoint theorem. Similarly, let mult be a term which satisfies

$$\langle x,y,z\rangle \in \mathsf{mult} \multimap (y = 0 \otimes z = 0) \oplus \exists y'.\exists z'.(y = \mathsf{S}(y') \otimes \langle x,z',z\rangle \in \mathsf{add} \otimes \langle x,y',z'\rangle \in \mathsf{mult}).$$

Certainly these terms satisfy the first and second conditions of representability:

Proposition 25
(a) $\langle \mathsf{n},\mathsf{m},\mathsf{k}\rangle \in \mathsf{add}$ *is provable in* ***SLST*** *iff* $n + m = k$;
(b) $\langle \mathsf{n},\mathsf{m},\mathsf{k}\rangle \in \mathsf{mult}$ *is provable in* ***SLST*** *iff* $n.m = k$.

We show now, by induction over N, that these terms represent addition and multiplication, respectively, with domains N and codomain N′.

Proposition 26. *The following are provable in* ***SLST***:
(a) $\forall x \in \mathsf{N}'.\forall y \in \mathsf{N}.\exists^! z \in \mathsf{N}'.(\langle x,y,z\rangle \in \mathsf{add})$;
(b) $\forall x.\forall y.(!(x \in \mathsf{N}) \otimes y \in \mathsf{N} \multimap \exists^! z.(z \in \mathsf{N}' \otimes \langle x,y,z\rangle \in \mathsf{mult}))$.

Proof. (a) We prove
(i) $\vdash \forall x \in \mathsf{N}'.\exists^! z \in \mathsf{N}'.(\langle x,0,z\rangle \in \mathsf{add})$, and
(ii) $\forall x \in \mathsf{N}'.\exists^! z \in \mathsf{N}'.(\langle x,y,z\rangle \in \mathsf{add}) \vdash \forall x \in \mathsf{N}'.\exists^! z \in \mathsf{N}'.(\langle x,\mathsf{S}y,z\rangle \in \mathsf{add})$.
An application of soft induction over N gives

$$y \in \mathsf{N} \vdash \forall x \in \mathsf{N}'.\exists^! z \in \mathsf{N}'.(\langle x,y,z\rangle \in \mathsf{add})$$

from which the desired conclusion trivially follows.

It is clear that $\langle x,0,x\rangle \in \mathsf{add}$ is provable. Suppose $\vdash \langle x,0,z\rangle \in \mathsf{add}$. Then $\vdash 0 = 0 \otimes x = z$ or $\vdash \exists y'.\exists z'.(0 = \mathsf{S}(y') \otimes z = \mathsf{S}(z') \otimes \langle x,y',z'\rangle \in \mathsf{add})$ is derivable. Since 0 is provably not the successor of any term, (i) follows.

For (ii), existence of an image follows from the following:

$$\dfrac{\dfrac{\dfrac{\dfrac{\langle x,y,z\rangle \in \mathsf{add} \vdash \langle x,y,z\rangle \in \mathsf{add} \quad \vdash \mathsf{S}y = \mathsf{S}y \otimes \mathsf{S}z = \mathsf{S}z}{\langle x,y,z\rangle \in \mathsf{add} \vdash \mathsf{S}y = \mathsf{S}y \otimes \mathsf{S}z = \mathsf{S}z \otimes \langle x,y,z\rangle \in \mathsf{add}}}{\langle x,y,z\rangle \in \mathsf{add} \vdash \exists y'.\exists z'.(\mathsf{S}y = \mathsf{S}y' \otimes \mathsf{S}z = \mathsf{S}z' \otimes \langle x,y,z\rangle \in \mathsf{add})}}{\langle x,y,z\rangle \in \mathsf{add} \vdash \langle x,\mathsf{S}y,\mathsf{S}z\rangle \in \mathsf{add}} \quad z \in \mathsf{N}' \vdash \mathsf{S}z \in \mathsf{N}'}{z \in \mathsf{N}' \otimes \langle x,y,z\rangle \in \mathsf{add} \vdash \mathsf{S}z \in \mathsf{N}' \otimes \langle x,\mathsf{S}y,\mathsf{S}z\rangle \in \mathsf{add}} \otimes\mathrm{R},\otimes\mathrm{L}$$

Here it is critical that we use the set N′, as we require that $z \in \mathsf{N}' \vdash \mathsf{S}z \in \mathsf{N}'$ is provable.

For uniqueness, see the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{w = \mathsf{S}w', w' = z \vdash w = \mathsf{S}z \quad \langle x,y,w'\rangle \in \mathsf{add} \vdash \langle x,y,w'\rangle \in \mathsf{add}}{w = \mathsf{S}w', \langle x,y,w'\rangle \in \mathsf{add}, \langle x,y,w'\rangle \in \mathsf{add} \multimap w' = z \vdash w = \mathsf{S}z}}{w = \mathsf{S}w', \langle x,y,w'\rangle \in \mathsf{add}, \forall w.(\langle x,y,w\rangle \in \mathsf{add} \multimap w = z) \vdash w = \mathsf{S}z}}{w = \mathsf{S}w' \otimes \langle x,y,w'\rangle \in \mathsf{add}, \forall w.(\langle x,y,w\rangle \in \mathsf{add} \multimap w = z) \vdash w = \mathsf{S}z}}{\exists w'.(w = \mathsf{S}w' \otimes \langle x,y,w'\rangle \in \mathsf{add}), \forall w.(\langle x,y,w\rangle \in \mathsf{add} \multimap w = z) \vdash w = \mathsf{S}z}}{\langle x,\mathsf{S}y,w\rangle \in \mathsf{add}, \forall w.(\langle x,y,w\rangle \in \mathsf{add} \multimap w = z) \vdash w = \mathsf{S}z}}{\forall w.(\langle x,y,w\rangle \in \mathsf{add} \multimap w = z) \vdash \langle x,\mathsf{S}y,w\rangle \in \mathsf{add} \multimap w = \mathsf{S}z)}}{\forall w.(\langle x,y,w\rangle \in \mathsf{add} \multimap w = z) \vdash \forall w.(\langle x,\mathsf{S}y,w\rangle \in \mathsf{add} \multimap w = \mathsf{S}z)}$$

Combining the last two results, we complete the proof of (ii). Applying soft induction yields the derivation of totality required.

(b) Similarly to the above, we can prove:

$$\vdash \exists^! z \in \mathsf{N}'.(\langle x,0,z\rangle \in \mathsf{mult}) \tag{1}$$

We can also prove

$$\exists^! z \in \mathsf{N}'.(\langle x,y,z\rangle \in \mathsf{mult}), \forall z \in \mathsf{N}'.\exists^! w \in \mathsf{N}'.(\langle x,z,w\rangle \in \mathsf{add}) \vdash \exists^! w \in \mathsf{N}'.(\langle x,\mathsf{S}y,w\rangle \in \mathsf{mult}) \tag{2}$$

From the representability of addition, we have

$$x \in \mathsf{N} \vdash \forall z \in \mathsf{N}'.\exists^! w \in \mathsf{N}'.(\langle z,x,w\rangle \in \mathsf{add}).$$

Hence we may derive

$$x \in \mathsf{N}, \exists^! z \in \mathsf{N}.(\langle x,y,z\rangle \in \mathsf{mult}) \vdash \exists^! w \in \mathsf{N}'.(\langle x,\mathsf{S}y,w\rangle \in \mathsf{mult}). \tag{3}$$

Applying soft induction to (1) and (3), we obtain

$$!(x \in \mathsf{N}), y \in \mathsf{N} \vdash \exists^! w \in \mathsf{N}'.(\langle x,y,w\rangle \in \mathsf{mult})$$

as required. □

Corollary 27. *Addition and multiplication of natural numbers are representable in **SLST** with domain* N *and codomain* N′.

Proof. The result follows immediately for multiplication, by an application of multiplexing to $(y \in \mathsf{N})$. For addition, we must first apply coercion to $(y \in \mathsf{N}')$, and then multiplexing to both arguments. □

There is a major difficulty with this approach, where we use N as a domain and N′ as a codomain; we do not have an obvious method for composing represented functions.[3] Thus we cannot infer representability of the polynomials from representability of addition and multiplication. To remedy this situation, we will go via a varaition on Lafont's representation of the polynomials in $\mathbf{SLL}_2$.

5.2. Polynomial functions and sets of preimages

Recall from the introduction that the typing of polynomial functions in $\mathbf{SLL}_2$ is somewhat eccentric; specifically, one cannot type the terms representing polynomial functions from N to N. This is also seemingly the case in **SLST**. For example, successor may be given as follows:

Lemma 28. *The following is provable in **SLST**:*

$$x \in \mathsf{N} \vdash \forall\alpha.(!\forall y(y \in \alpha \multimap \mathsf{S}y \in \alpha) \otimes \forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha) \multimap (0 \in \alpha \multimap \mathsf{S}x \in \alpha))$$

We will give the set

$$\{\, x \mid \alpha.(!\forall y(y \in \alpha \multimap \mathsf{S}y \in \alpha) \otimes \forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha) \multimap (0 \in \alpha \multimap x \in \alpha)) \,\}$$

the name $\mathsf{N}\langle X+1\rangle$. This notation comes from a similar structure in $\mathbf{SLL}_2$:

Definition 29. We extend the definition A^n to *polynomial expressions* as follows:

$$A^X = !A \qquad A^{P+Q} = A^P \otimes A^Q \qquad A^{PQ} = (A^P)^Q.$$

Given a polynomial expression P, we write $A\langle P\rangle$ for the formula A where each subformula $!B$ is replaced by B^P.

[3] This is not the issue with composition mentioned in Section 4; however, note that we have not yet proven multiplication to be 0-representable.

It is easy to see that the following generalized rules are derivable:

$$\frac{\Gamma \vdash A}{\Gamma^P \vdash A^P}\, SP(P) \qquad \frac{\Gamma, A^{P(n)} \vdash C}{\Gamma, A^P \vdash C}\, mplx(P)$$

where Γ^P is defined pointwise on the elements of Γ.

This scheme allows Lafont to define a representation of addition in $\mathbf{SLL}_2$:

$$\mathsf{N}, \mathsf{N} \vdash \mathsf{N}\langle X + X\rangle,$$

or more generally

$$\mathsf{N}\langle P\rangle, \mathsf{N}\langle Q\rangle \vdash \mathsf{N}\langle P + Q\rangle.$$

To annotate this proof with set theoretic information, so that it yields a proof of the totality of addition in **SLST**, we would need to give an operation "+" on all terms of **SLST**, such that

(a) $t + 0 = t$, and
(b) $t + \mathsf{S}s = \mathsf{S}(t + s)$

However, such operations do not fit naturally into a set theoretic setting (where we should associate functions with their graphs), so instead we work with a term inspired by the "Types with integer" approach of Baillot and Mogbil.

Lemma 30. *Consider the following term of* ***SLST***:

$$\mathsf{N}\langle P + Q\rangle[\mathsf{add}] := \{\, t \mid t = \langle x,y\rangle \otimes \forall\alpha.((\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha))^P \otimes (\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha))^Q \multimap (0 \in \alpha \multimap \exists^! z.(z \in \alpha \otimes \langle x,y,z\rangle \in \mathsf{add}))) \,\}$$

The following is provable in ***SLST***:

$$x \in \mathsf{N}\langle P\rangle, y \in \mathsf{N}\langle Q\rangle \vdash \langle x,y\rangle \in \mathsf{N}\langle P + Q\rangle[\mathsf{add}]$$

Proof. See appendices.

We will call the term $\mathsf{N}\langle P + Q\rangle[\mathsf{add}]$ a set of add preimages, the idea being that any pair $\langle x,y\rangle$ provably in $\mathsf{N}\langle P + Q\rangle[\mathsf{add}]$ has a unique sum in any set containing 0 and closed under successor. Similarly:

Lemma 31. *Consider the following term of* ***SLST***:

$$\mathsf{N}\langle PQ\rangle[\mathsf{mult}] := \{\, t \mid t = \langle x,y\rangle \otimes \forall\alpha.((\forall y(y \in \alpha \multimap \mathsf{S}y \in \alpha))^{PQ} \multimap (0 \in \alpha \multimap \exists^! z.(z \in \alpha \otimes \langle x,y,z\rangle \in \mathsf{mult}))) \,\}$$

The following is provable in ***SLST****:*

$$x \in \mathsf{N}\langle P\rangle, y \in \mathsf{N}\langle Q\rangle \vdash \langle x,y\rangle \in \mathsf{N}\langle PQ\rangle[\mathsf{mult}]$$

More generally, given a polynomial expression P and a term t of **SLST**, define the following term:

$$\mathsf{N}\langle P\rangle[t] := \{\, x \mid \forall\alpha.(\forall y(y \in \alpha \multimap \mathsf{S}y \in \alpha)^P \multimap (0 \in \alpha \multimap \exists^! z.(z \in \alpha \otimes \langle x,z\rangle \in t)) \,\}$$

Define also the *pseudo-degree* δP of a polynomial expression P as follows:

$$\delta n = 0, \quad \delta X = 1, \quad \delta(P + Q) = \delta(PQ) = \delta P + \delta Q.$$

Theorem 32. *For any polynomial expression P, there exists a term p of* ***SLST*** *such that*

(a) $(x \in \mathsf{N})^{(\delta P)} \vdash x \in N\langle P\rangle[p]$ *is provable in* ***SLST***
(b) $\vdash \langle a,b\rangle \in p$ *is provable in* ***SLST*** *if and only if, for some $n,m \in \mathbb{N}$, $\vdash a = \mathsf{n}$, $\vdash b = \mathsf{m}$, and $P(n) = m$.*

Proof. By induction on the structure of P. If P is a constant n then we have $\delta P = 0$ and $\vdash \forall\alpha.(\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha)^n \multimap (0 \in \alpha \otimes \exists^! z.(z \in \alpha \otimes \langle x,z\rangle \in \{\, \langle x,z\rangle \mid z = \mathsf{n} \,\})))$. Suppose now that for polynomial expressions containing less than m instances of $+$ and $\times$, the theorem holds. Let P contain m constructors, and be of the form $Q + R$. Then Q and R satisfy the conditions of the

induction hypothesis, and there are terms q and r such that $(x \in \mathsf{N})^{(\delta Q)} \vdash x \in N\langle Q\rangle[q]$ and $(x \in \mathsf{N})^{(\delta R)} \vdash x \in N\langle R\rangle[r]$. As shown in Prop 55,

$$x \in \mathsf{N}\langle Q\rangle[q], y \in \mathsf{N}\langle R\rangle[r] \vdash ((y \in \alpha \multimap \mathsf{S}y \in \alpha)^{P+Q} \multimap (0 \in \alpha \multimap A))$$

where $A = \exists^!u.\exists^!v.\exists^!w(w \in \alpha \otimes \langle x,u\rangle \in t \otimes \langle x,v\rangle \in s \otimes \langle u,v,w\rangle \in \mathsf{add}))$ and $\exists^!u.\exists^!v.(\langle \mathsf{n},u\rangle \in t \otimes \langle \mathsf{n},v\rangle \in s \otimes \langle u,v,w\rangle \in \mathsf{add})))$ is provable iff w is k, where $k = P(n)$. The application of two cuts gives

$$(x \in \mathsf{N})^{(\delta Q)} \vdash , (x \in \mathsf{N})^{(\delta R)} \vdash ((y \in \alpha \multimap \mathsf{S}y \in \alpha)^{P+Q} \multimap (0 \in \alpha \multimap A))$$

completing the proof. The case for multiplication is similar. □

The formula $x \in \mathsf{N}\langle P\rangle[t]$ is powerful because it contains information about the totality of t, but also has computational content. For instance, we can perform induction over $\mathsf{N}\langle P\rangle[t]$:

Proposition 33. *The following inference is derivable in* ***SLST***:

$$\frac{\Gamma \vdash A[x := 0] \qquad \Delta, A[x := y] \vdash A[x := \mathsf{S}y]}{\Gamma, \Delta^P, s \in \mathsf{N}\langle P\rangle[t] \vdash \exists^!w(A[x := w] \otimes \langle s,w\rangle \in t)}\ \mathsf{N}\langle P\rangle[t] - ind.$$

Proof. First note that, since $w \in \{x \mid A\} \circ\!\!-\!\!\circ A[x := w]$, it is easy to derive, in **SLST**, that $\exists^!w(w \in \{x \mid A\} \otimes \langle s,w\rangle \in t) \vdash \exists^!w(A[x := w] \otimes \langle s,w\rangle \in t)$. Using that fact, the following is a derivation of the proposition:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Delta, A[x := y] \vdash A[x := \mathsf{S}y]}{\Delta, y \in \{x \mid A\} \vdash \mathsf{S}y \in \{x \mid A\}} \in \mathrm{L}, \in \mathrm{R}}{\Delta \vdash y \in \{x \mid A\} \multimap \mathsf{S}y \in \{x \mid A\}} \multimap \mathrm{R}}{\Delta \vdash \forall y.(y \in \{x \mid A\} \multimap \mathsf{S}y \in \{x \mid A\})} \forall \mathrm{L}}{\Delta^P \vdash (\forall y.(y \in \{x \mid A\} \multimap \mathsf{S}y \in \{x \mid A\}))^P} \mathrm{SP(P)} \qquad \dfrac{\dfrac{\Gamma \vdash A[x := 0]}{\Gamma \vdash 0 \in \{x \mid A\}} \in R \qquad \exists^!w(w \in \{x \mid A\} \otimes \langle s,w\rangle \in t) \vdash \exists^!w(A[x := w] \otimes \langle s,w\rangle \in t)}{\Gamma, 0 \in \{x \mid A\} \multimap \exists^!w(w \in \{x \mid A\} \otimes \langle s,w\rangle \in t) \vdash \exists^!w(A[x := w] \otimes \langle s,w\rangle \in t)} \multimap \mathrm{L}}{\Gamma, \Delta^P, (\forall y.(y \in \{x \mid A\} \multimap \mathsf{S}y \in \{x \mid A\}))^P \multimap (0 \in \{x \mid A\} \multimap \exists^!w(w \in \{x \mid A\} \otimes \langle s,w\rangle \in t)) \vdash \exists^!w(A[x := w] \otimes \langle s,w\rangle \in t)} \multimap \mathrm{L}$$

$$\frac{\Gamma, \Delta^P, (\forall y.(y \in \{x \mid A\} \multimap \mathsf{S}y \in \{x \mid A\}))^P \multimap (0 \in \{x \mid A\} \multimap \exists^!w(w \in \{x \mid A\} \otimes \langle s,w\rangle \in t)) \vdash \exists^!w(A[x := w] \otimes \langle s,w\rangle \in t)}{\Gamma, \Delta^P, s \in \mathsf{N}\langle P\rangle[t] \vdash \exists^!w(A[x := w] \otimes \langle s,w\rangle \in t)} \forall \mathrm{L}.$$ □

Corollary 34. *Each polynomial is* 0*-representable in* ***SLST***.

Proof. Let P be a polynomial expression. Then we know, from Theorem 32, that for some n, there exists a term p such that p satisfies the second condition of 0-representation (32(b)) and $(s \in \mathsf{N})^{(n)} \vdash s \in N\langle P\rangle[p]$ is provable in **SLST** (32(a)) . Now apply $\mathsf{N}\langle P\rangle[p]$ induction to the formula $x \in \mathsf{N}'$, to obtain

$$t \in \mathsf{N}\langle P\rangle[p] \vdash \exists^!w(w \in N' \otimes \langle t,w\rangle \in p).$$

Apply cut to obtain

$$(t \in \mathsf{N})^{(n)} \vdash \exists^!w(w \in N' \otimes \langle t,w\rangle \in p).$$

6. Words over a finite alphabet

In this section we consider the representation of binary words in **SLST**, as a special case of words over n symbols. As one might expect, a similar separation occurs for the words as occurs for the natural numbers. First, define

$$\varepsilon := \emptyset, \quad \mathsf{S}_i(t) := \langle \mathsf{i},t\rangle.$$

The following two definitions each give a term which represents the words over an alphabet with 2 elements:

Definition 35 (*Soft Words*).

$$\mathsf{W}_2 = \{x \mid \forall\alpha(!\forall y(y \in \alpha \multimap \mathsf{S}_0y \in \alpha) \multimap !\forall y(y \in \alpha \multimap \mathsf{S}_1y \in \alpha) \multimap (\varepsilon \in \alpha \multimap x \in \alpha))\}$$

Definition 36 (*Safe Words*).

$$x \in \mathsf{W}_2' \circ\!\!-\!\!\circ x = \varepsilon \oplus \exists y(y \in \mathsf{W}_2' \otimes x = \mathsf{S}_0y) \oplus \exists y(y \in \mathsf{W}_2' \otimes x = \mathsf{S}_1y)$$

where the existence of W'_2 is given by the fixpoint theorem. Define W_n,W'_n similarly. From this point onward, let $\mathbb{W}$ stand for $\mathbb{W}_2$, the set (in the usaul sense) of binary words, and similarly let $\mathsf{W} := \mathsf{W}_2$ and $\mathsf{W}' := \mathsf{W}'_2$.

We may derive an induction principle over the structure of strings in W_n, similar to that for soft naturals:

Proposition 37. *The following inference is derivable in* ***SLST****:*

$$\frac{\Gamma \vdash A[x := \varepsilon] \qquad \Delta_0, A[x := y] \vdash A[x := \mathsf{S}_0 y] \quad \ldots \quad \Delta_{n-1}, A[x := y] \vdash A[x := \mathsf{S}_{n-1} y]}{\Gamma, !\Delta, s \in \mathsf{W}_n \vdash A[x := s]}\ \mathsf{W}_n - ind.$$

Corollary 38. *For each $n \leq m$, and for any p,*

$$x \in \mathsf{W}_n \vdash !^p x \in \mathsf{W}'_m.$$

We may capture the length function $|x|$ as follows:

Proposition 39. *Let the term len_n be defined by fixpoint as*

$$\langle x,y\rangle \in \mathsf{len} \multimap (x = \varepsilon \otimes y = 0) \oplus \exists x'.\exists y'.((x = \mathsf{S}_0(x') \oplus \cdots \oplus x = \mathsf{S}_{n-1}(x')) \otimes y = \mathsf{S}(y') \otimes \langle x',y'\rangle \in \mathsf{len}_n).$$

Then the following is provable in ***SLST****:*

$$x \in \mathsf{W}_n \vdash x \in \mathsf{N}\langle X^n\rangle[\mathsf{len}_n].$$

We leave the proof as an easy exercise.

The purpose of all this is to provide a polynomial bound on the output of a Turing machine; as such, the following is an important but trivial generalisation of the preceding proposition:

Proposition 40. *Given a term p representing a polynomial expression P, let p' be defined as $\{\ \langle x,w\rangle \mid \exists^! v.(\langle x,v\rangle \in \mathsf{len}_n \otimes \langle v,w\rangle \in p)\ \}$. Writing $P(Q)$ for the polynomial expression given by replacing each instance of X with Q, we have*

$$(x \in \mathsf{W}_n)^{\delta P} \vdash x \in N\langle P(X^n)\rangle[p'].$$

Meanwhile, the safe words are well behaved with respect to the successor functions.

Proposition 41. *For each $i < n$*

$$x \in \mathsf{W}'_n \vdash \mathsf{S}_i x \in \mathsf{W}'_n$$

is provable in ***SLST****.*

Corollary 42. *The successor functions on $\mathbb{W}_n$ are 0-representable with multiplicity 1 from W'_n to W'_n.*

Additionally, one may define functions by cases of a term in W'_n:

Proposition 43. *Given functions $\psi_\varepsilon : T \to U$ and $\psi_i : \mathbb{W}_n \times T \to U$, define a new function $\phi : \mathbb{W}_n \times T \to U$ as follows*:

$$\phi(\varepsilon,x) = \psi_\varepsilon(x);$$
$$\phi(i.w,x) = \psi_i(w,x).$$

Suppose now that T and U are represented by terms t and u, and that ψ_ε is 0-representable from t to u by h_ε, and ψ_i is 0-representable from W',t to u by h_i, such that

(a) *The multiplicity of W' in each h_i is 1, and*
(b) *The multiplicity of t in h_ε, and in each h_i, is some value r.*

Then ϕ is 0-representable with domains W',t and codomain u.

Proof. Define the **SLST** term f as follows:

$$\langle x,y,z\rangle \in f \multimap (x = \varepsilon \otimes \langle y,z\rangle \in h_\varepsilon) \oplus \exists x'(x = \mathsf{S}_0(x') \otimes \langle x',y,z\rangle \in h_0) \oplus \cdots \oplus \exists x'(x = \mathsf{S}_{n-1}x' \otimes \langle x',y,z\rangle \in h_{n-1}).$$

Table 2
SLC typing rules, plus typing for comprehension

$$\frac{}{\mathtt{x} : A \vdash \mathtt{x} : A}\ \mathrm{Ax}$$

$$\frac{\Gamma,\mathtt{x} : A \vdash \mathtt{M} : B}{\Gamma \vdash \lambda\mathtt{x}.\mathtt{M} : A \multimap B}\ \multimap\mathrm{R} \qquad \frac{\Gamma \vdash \mathtt{N} : A \quad \Delta,\mathtt{x} : B \vdash \mathtt{M} : C}{\Gamma,\Delta,\mathtt{y} : A \multimap B \vdash \mathtt{M}[\mathtt{x} := \mathtt{y}\mathtt{N}] : C}\ \multimap\mathrm{L}$$

$$\frac{\mathtt{x}_1 : A_1,\ldots,\mathtt{x}_n : A_n \vdash \mathtt{M} : C}{\mathtt{y}_1 : !A_1,\ldots,\mathtt{y}_n : !A_n \vdash \mathit{let}\ \vec{\mathtt{y}}\ \mathit{be}\ !\vec{\mathtt{x}}\ \mathit{in}\ !\mathtt{M} : !C}\ \mathrm{SP} \qquad \frac{\Gamma,\mathtt{x}_1 : A \ldots \mathtt{x}_n : A \vdash \mathtt{M} : C}{\Gamma,\mathtt{y} : !A \vdash \mathit{let}\ \mathtt{y}\ \mathit{be}\ !\mathtt{x}\ \mathit{in}\ \mathtt{M}[\mathtt{x}_1 := \mathtt{x},\ldots,\mathtt{x}_n := \mathtt{x}] : C}\ \mathrm{mplx}$$

$$\frac{\mathtt{x} : A[x := t],\Gamma \vdash \mathtt{M} : C}{\mathtt{x} : \forall x.A,\Gamma \vdash \mathtt{M} : C}\ \forall\mathrm{L} \qquad \frac{\Gamma \vdash \mathtt{M} : C}{\Gamma \vdash \mathtt{M} : \forall x.C}\ \forall\mathrm{R}$$

$$\frac{\mathtt{x} : A[x := t],\Gamma \vdash \mathtt{M} : C}{\mathtt{x} : t \in \{x|A\},\Gamma \vdash \mathtt{M} : C}\ \in\mathrm{L} \qquad \frac{\Gamma \vdash \mathtt{M} : C[x := t]}{\Gamma \vdash \mathtt{M} : t \in \{x|A\}}\ \in\mathrm{R}$$

$$\frac{\Gamma \vdash \mathtt{N} : A \quad \Gamma',x : A \vdash \mathtt{M} : C}{\Gamma,\Gamma' \vdash \mathtt{M}[x := \mathtt{N}] : C}\ \textsc{Cut}$$

By assumption, $(y \in t)^r \vdash \exists^! z.(\langle y,z\rangle \in h_\varepsilon)$, from which we derive

$$x = \varepsilon,(y \in t)^r \vdash \exists^! z.(\langle x,y,z\rangle \in f).$$

Also, for each $0 \le i \le n-1$, we have

$$x' \in \mathsf{W}'_n \otimes x = \mathsf{S}_i(x'),(y \in t)^r \vdash \exists^! z.(\langle x,y,z\rangle \in f).$$

Hence we have

$$x \in \mathsf{W}'_n,(y \in t)^r \vdash \exists^! z.(\langle x,y,z\rangle \in f).$$

Corollary 44. *The predecessor function on* $\mathbb{W}$ *is* 0*-representable with both domain and codomain* W'_n*, and multiplicity* 1.

6.1. Soft lambda calculus and polynomial soundness

We will demonstrate in the next section that any function computable in polynomial time is 0-representable, but first we address the issue of "polytime soundness"—that is, we must verify that any 0-representable function is polytime computable. To do so, we turn to the Soft lambda-calculus of Baillot and Mogbil [2]. Soft lambda-calculus (**SLC**) is a calculus typable in Soft *Affine* Logic—that is, **SLL** with unrestricted weakening.

We give the typing rules for Soft Lambda calculus in Table 2.

A typed term of **SLC** is a pair $\mathtt{M} : A$ arising from a judgement $\Gamma \vdash \mathtt{M} : A$; such a term $\mathtt{M}$ is a special case of a well-formed term.[4] Given such a term, we define its *depth* and *size* as follows:

Definition 45

(a) The *size* $|\mathtt{M}|$ of a term $\mathtt{M}$ is given by

$$|\mathtt{x}| = 1, \qquad |\lambda\mathtt{x}.\mathtt{M}| = |\mathtt{M}| + 1, \qquad |(\mathtt{M}\mathtt{N})| = |\mathtt{M}| + |\mathtt{N}|$$
$$|!\mathtt{M}| = |\mathtt{M}| + 1 \qquad |\mathit{let}\ \mathtt{M}\ \mathit{be}\ !\mathtt{x}\ \mathit{in}\ \mathtt{N}| = |\mathtt{M}| + |\mathtt{N}| + 1$$

(b) The *depth* of a term $\mathtt{M}$ is defined as follows: let $\mathtt{N}$ be a subterm of $\mathtt{M}$. Define $d(\mathtt{N},\mathtt{M})$ to be the number of subterms $\mathtt{L}$ of $\mathtt{M}$ such that $\mathtt{N}$ is a subterm of $\mathtt{L}$ and $\mathtt{L}$ is of the form $!\mathtt{L}'$. The depth $d(\mathtt{M})$ of $\mathtt{M}$ is then the maximum value of $d(\mathtt{N},\mathtt{M})$ for $\mathtt{N}$ a subterm of $\mathtt{M}$.

[4] The typed/typable terms are not the only terms of interest in **SLC**; the untyped calculus also enjoys polynomial reduction.

Table 3
Fixpoint typing rules

$$\frac{\Gamma,\mathtt{x} : t \in \mu X.A \vdash \mathtt{M} : B}{\Gamma,\mathtt{y} : A[X := \mu X.A, z := t] \vdash \mathtt{M}[\mathtt{x} := \mathtt{fold}\ \mathtt{y}] : B}\ (\text{left unfold}) \qquad \frac{\Gamma \vdash \mathtt{M} : \mu X.A}{\Gamma \vdash \mathtt{unfold}\ \mathtt{M} : A[X := \mu X.A]}\ (\text{right unfold})$$

$$\frac{\Gamma,\mathtt{x} : A[X := \mu X.A, z := t] \vdash \mathtt{M} : B}{\Gamma,\mathtt{y} : t \in \mu X.A \vdash \mathtt{M}[\mathtt{x} := \mathtt{unfold}\ \mathtt{y}] : B}\ (\text{left fold}) \qquad \frac{\Gamma \vdash \mathtt{M} : A[X := \mu X.A, z := t]}{\Gamma \vdash \mathtt{fold}\ \mathtt{M} : t \in \mu X.A}\ (\text{right fold})$$

The reductions rules of **SLC** are the following

$$\begin{aligned}
(\beta):&\ ((\lambda \mathtt{x} \mathtt{M}.)\ \mathtt{N}) \longrightarrow \mathtt{M}[\mathtt{x} := \mathtt{N}];\\
(!):&\ \mathit{let}\ !\mathtt{N}\ \mathit{be}\ !\mathtt{x}\ \mathit{in}\ \mathtt{M} \longrightarrow \mathtt{M}[\mathtt{x} := \mathtt{N}];\\
(\mathrm{com1}):&\ \mathit{let}\ (\mathit{let}\ \mathtt{M}_1\ \mathit{be}\ !\mathtt{y}\ \mathit{in}\ \mathtt{M}_2)\ \mathit{be}\ !\mathtt{x}\ \mathit{in}\ \mathtt{M}_3 \longrightarrow \mathit{let}\ \mathtt{M}_1\ \mathit{be}\ !\mathtt{y}\ \mathit{in}\ (\mathit{let}\ \mathtt{M}_2\ \mathit{be}\ !\mathtt{x}\ \mathit{in}\ \mathtt{M}_3);\\
(\mathrm{com2}):&\ (\mathit{let}\ \mathtt{M}\ \mathit{be}\ !\mathtt{x}\ \mathit{in}\ \mathtt{M}_2)\ \mathtt{M}_3 \longrightarrow \mathit{let}\ \mathtt{M}_1\ \mathit{be}\ !\mathtt{x}\ \mathit{in}\ (\mathtt{M}_2\mathtt{M}_3).
\end{aligned}$$

We have the following theorem:

Theorem 46 (Polytime strong normalization). *For any integer d there is a polynomial P_d (with degree linear in d) such that for any term $\mathtt{M}$ of depth d, any sequence of reductions of t has length bounded by $P_d(|\mathtt{M}|)$.*

Since this polytime normalization theorem holds even for the type-free calculus, **SLC** may be extended with recursive types (fixpoints); Baillot and Mogbil thus extend their typed calculus to a calculus with fixpoints (**ISALF**) while retaining polytime normalization. In our setting, we also have access to fixpoints, but they are derivable. The (derivable) typing for set theoretic fixpoints differs from that for **ISALF**; we show the types of the derivations in Table 3.
In this table, the abbreviations

$$\mathtt{fold}\ \mathtt{M} := \lambda \mathtt{yzw.yzw}\ (\lambda \mathtt{v.v}\ \mathtt{M})$$

and

$$\mathtt{unfold}\ \mathtt{N} := \mathtt{N}(\lambda \mathtt{v.v}\ \lambda \mathtt{w.w}\ \lambda \mathtt{xy.y})$$

are derived from the definitions in the fixpoint theorem.

Theorem 47 (Subject reduction). *If we have $\Gamma \vdash \mathtt{M} : A$ in **SLC** , and $\mathtt{M} \to \mathtt{M}'$, then $\Gamma \vdash \mathtt{M}' : A$.*

We now use this calculus to help demonstrate polynomial soundness. Observe that we may translate any proof in **SLST** into a typing judgement in **SLC**—instances of nullary multiplexing are replaced by first a weakening and then a unary multiplexing, and then all the missing connectives (including the additive $\oplus$) may be defined, since we have access to unrestricted weakening. In particular, note that the existential is given by

$$\exists y.A := \forall x.(\forall y.(A \multimap t_0 \in x) \multimap t_0 \in x),$$

multiplicative conjunction by

$$A \otimes B := \forall x.((A \multimap t_0 \in x) \multimap (B \multimap t_0 \in x) \multimap t_0 \in x).$$

and additive disjunction by

$$A \oplus B := \forall x.(A \multimap B \multimap t_0 \in x) \multimap t_0 \in x).$$

(where t_0 is an arbitrary closed term) with the standard lambda terms to represent constructs such as `inl` ,`inr` multiplicative pairing (written $- \otimes -$), and projections `fst` and `snd`.

We now give canonical proofs that, for any word $w \in \mathbb{W}$, $\mathsf{w} \in \mathsf{W}$ and $\mathsf{w} \in \mathsf{W}'$:

Definition 48. Let $w := i_0 \cdots i_n \in \mathbb{W}_2$. Then $\bar{w}$ denotes

$$\lambda \mathtt{x}_0\mathtt{x}_1.(\mathit{let}\ \mathtt{x}_0\ \mathit{be}\ !\mathtt{z}_0\ \mathit{in}\ (\mathit{let}\ \mathtt{x}_1\ \mathit{be}\ !\mathtt{z}_1\ \mathit{in}\ (\lambda \mathtt{y}.(\mathtt{z}_{i_0} \cdots (\mathtt{z}_{i_n})))))).$$

We use $\hat{\varepsilon}$ to denote $\mathtt{fold\ inl}\lambda\mathtt{x.x}$ and $\hat{w.i}$ to denote

$$\mathtt{fold\ inr}(\lambda\mathtt{z.z}(\hat{w} \otimes \hat{i}))$$

where $\hat{0} := \mathtt{inl}\lambda x.x$ and $\hat{1} := \mathtt{inr}\lambda x.x$.

A W representation of w is a term M of **SLC** such that $\vdash M : (\mathtt{w} \in \mathsf{W})$. A W′ representation of w is a term M of **SLC** such that $\vdash M : (\mathtt{w} \in \mathsf{W}')$.

Now define the relation $\approx$ on terms of **SLC** as the least binary congruence satisfying:

$$\begin{aligned}
(\eta): &\quad \lambda\mathtt{x.Mx} \approx \mathtt{M}, \text{if } \mathtt{x} \notin FV(\mathtt{M});\\
(\text{let}): &\quad \mathit{let}\ \mathtt{N}\ \mathit{be}\ !\mathtt{x}\ \mathit{in}\ \mathtt{M} \approx \mathtt{M}, \text{ if } x \notin FV(\mathtt{M});\\
(\lambda - \text{let}): &\quad \lambda\mathtt{x.}(\mathit{let}\ \mathtt{M}\ \mathit{be}\ !\mathtt{y}\ \mathit{in}\ \mathtt{N}) \approx \mathit{let}\ \mathtt{M}\ \mathit{be}\ !\mathtt{y}\ \mathit{in}\ \lambda\mathtt{x.N}, \text{ if } \mathtt{x} \notin FV(\mathtt{M});\\
(\text{let–let}): &\quad \mathit{let}\ \mathtt{M}\ \mathit{be}\ !\mathtt{x}\ \mathit{in}\ (\mathit{let}\ \mathtt{N}\ \mathit{be}\ !\mathtt{y}\ \mathit{in}\ \mathtt{L}) \approx \mathit{let}\ \mathtt{N}\ \mathit{be}\ !\mathtt{y}\ \mathit{in}\ (\mathit{let}\ \mathtt{M}\ \mathit{be}\ !\mathtt{x}\ \mathit{in}\ \mathtt{L}).
\end{aligned}$$

It is easy to see that $\approx$ is compatible with $\longrightarrow^*$. That is, if $\mathtt{M} \approx \mathtt{N}$ and $\mathtt{M} \longrightarrow^* \mathtt{M}'$, then there is a term $\mathtt{N}'$ such that $\mathtt{N} \longrightarrow^* \mathtt{N}'$ and $\mathtt{N} \approx \mathtt{N}'$.

Lemma 49

(a) $\bar{w}$ *is a* W *representation of* w;
(b) *If* M *is a* W *representation of* w, *then* $M \approx \bar{w}$;
(c) $\hat{w}$ *is a* W′ *representation of* w;
(d) *If* N *is a* W′ *representation of* w, *then* $N \approx \hat{w}$.

Now suppose that we have some statement of the representability of a function $\phi : \mathbb{W}_2 \to \mathbb{W}_2$. Then we have

$$\vdash G : \forall x(!(x \in \mathsf{W}_2) \multimap \exists y(y \in \mathsf{W}_2' \otimes \langle x,y\rangle \in f))$$

as the result of a typing derivation in **SLC**. Let $w \in \mathbb{W}_2$. Then $\vdash \bar{w} : \mathtt{w} \in \mathsf{W}_2$ is derivable. In addition, we have $\vdash G : !(\mathtt{w} \in \mathsf{W}_2) \multimap \exists y.(y \in \mathsf{W}_2'. \otimes \langle \mathtt{w},y\rangle \in f))$, so

$$\vdash G!\bar{w} : \exists y.(y \in \mathsf{W}_2' \otimes \langle \mathtt{w},y\rangle \in f).$$

By subject reduction the normal form of $G!\bar{w}$ also has this type, and must therefore be of the form $\lambda\mathtt{x.x}(\lambda\mathtt{v.vNL})$. Moreover, $\vdash \mathtt{N} : u \in \mathsf{W}_2'$ and $\vdash \mathtt{L} : \langle \mathtt{w},u\rangle \in f$ must be derivable for some term u of **SLST**. Hence u is $\hat{w}'$ for some word $w' \in \mathbb{W}_2$. Finally, we obtain, setting $\mathtt{id} := \lambda\mathtt{x.x}$ and $\mathtt{fst} := \lambda\mathtt{xy.x}$,

$$\begin{aligned}
&\lambda\mathtt{z.}(((G\ \mathtt{z})\ \mathtt{id})\ \mathtt{fst})\ !\bar{w} \longrightarrow ((G\ !\bar{w})\ \mathtt{id})\ \mathtt{fst})\\
&\qquad \longrightarrow^* ((\lambda\mathtt{x.x}\ (\lambda\mathtt{v.vNL}))\ \mathtt{id})\ \mathtt{fst} \longrightarrow^* \lambda\mathtt{v.vNL}\ \mathtt{fst} \longrightarrow^* \mathtt{N} \approx \hat{w}',
\end{aligned}$$

as required.

Theorem 50. *Representable functions are polytime computable.*

Proof. Given a word w, its canonical representant $\bar{w}$ has depth one, and so the depth of $\lambda\mathtt{z.}(((G\ \mathtt{z})\ \mathtt{id})\ \mathtt{fst})\ !\bar{w}$ is a constant d no matter which word we pick. The size $|\bar{w}|$ is $10 + |w|$; let the size of $\lambda z.(((G\ z)\ \mathrm{id})\ \mathrm{fst})$ be n. We have, by polytime strong normalization, a bounding function $P_d(n + 10 + |w|)$—a polynomial in $|w|$.

7. Simulation of Turing machines

We present an encoding of single tape polynomial-time Turing machines in **SLST**, demonstrating that the latter proves total any function computable in polynomial time.

We will work with Turing machines over a three letter alphabet (1, 0 and b = "blank") with set of states $Q_n = \{q_0, \ldots, q_{n-1}\}$, where q_0 is the initial state. The current configuration of the machine may then be given as a triple $\langle q,l,r\rangle \in \mathrm{Conf} = Q \times \mathbb{W}_3 \times \mathbb{W}_3$, where q is the current state, l is the non-blank portion of the tape to the left of the head, and r is the non-blank portion of the tape to the right of the head. By convention, l is written in reverse order, and r includes the symbol currently read.

Definition 51. A function $\phi : \mathbb{W}_2 \to \mathbb{W}_2$ is a polynomial-time function if there is some Turing machine T and some polynomial P such that after running T with input the string x for $P(|x|)$ steps, the output (the non-empty right-hand portion of the tape) is $\phi(x)$.

We show now how, given such a function ϕ, one may construct a term f that represents it in **SLST**.

The set of states of T may be represented in **SLST** by the term $\mathsf{Q}_n = \{0,\dots,n-1\}$, with evident bijection. We represent the set of possible configurations of T by the term

$$\mathsf{Conf} = \mathsf{Q}_n \times \mathsf{W}_3' \times \mathsf{W}_3'.$$

By Corollary 38, $x \in \mathsf{W}_2 \vdash x \in \mathsf{W}_3'$. It is clear that $x \in W_2 \vdash \langle 0,\varepsilon,x\rangle \in \mathsf{Conf}$ is provable in **SLST**: an application of cut gives:

$$x \in \mathsf{W}_2 \vdash \langle 0,\varepsilon,x\rangle \in \mathsf{Conf} \tag{4}$$

The transition function for T may be expressed as a function $\delta : \mathsf{Conf} \to \mathsf{Conf}$: given a particular state and a particular read symbol, the new tape is given by successor and predecessor operations on the left and right tapes. Recall that successor and predecessor are both 0-representable from W_3' to W_3' with multiplicity 1. Since the transition function is defined by a conditional on W_3' over functions satisfying the conditions of Proposition 43, it is 0-representable with domain Conf, codomain Conf and multiplicity 1. Let b be a term of **SLST** representing this function.

Define the **SLST** term d by fixpoint as follows:

$$\langle t,w\rangle \in d \multimap (t = 0 \otimes w = \langle 0,\varepsilon,c\rangle) \tag{5}$$
$$\oplus\, \exists t'\exists x\exists x'.\exists y\exists y'.\exists z\exists z'.(w = \langle x,y,z\rangle \tag{6}$$
$$\otimes\, \langle\langle x',y',z'\rangle,w\rangle \in b \otimes t = \mathsf{S}t' \otimes \langle t',\langle x',y',z'\rangle\rangle \in d) \tag{7}$$

This term represents the extended transition function of the machine T with initial configuration c.

Given a polynomial P, we want to know what the configuration of the machine is after $P(x)$ steps—the function $\psi(P(x))$. To arrive at this we use induction over $\mathsf{N}\langle P(X^2)\rangle[p']$, as defined in Proposition 33, where, as before, $p := \{x \mid \exists^! v.(\langle x,v\rangle \in \mathsf{len}_n \otimes \langle v,w\rangle \in p)\}$:

$$\frac{c \in \mathsf{Conf} \vdash \exists^! c.(c \in \mathsf{Conf} \otimes \langle 0,c\rangle \in d) \quad \exists^! c.(c \in \mathsf{Conf} \otimes \langle y,c\rangle \in d) \vdash \exists^! c.(c \in \mathsf{Conf} \otimes \langle \mathsf{S}y,c\rangle \in d)}{c \in \mathsf{Conf}, x \in \mathsf{N}\langle P(X^2)\rangle[p'] \vdash \exists^! k.(k \in \mathsf{Conf} \otimes \exists^! n.(\langle n,k\rangle \in d) \otimes \langle x,n\rangle \in p')}\ \mathsf{N}\langle P\rangle[t] - ind. \tag{8}$$

From Lemma 40:

$$(x \in \mathsf{W}_2)^{\delta P} \vdash x \in \mathsf{N}\langle P(X^2)\rangle[p']. \tag{9}$$

Combining (4), (8) and (9), we obtain

$$(x \in \mathsf{W}_2)^{1+\delta P}.x \in \mathsf{W}_3' \vdash \exists^! k.(k \in \mathsf{Conf} \otimes \exists^! n.(\langle n,k\rangle \in d) \otimes \langle x,n\rangle \in p') \tag{10}$$

Finally, we extract the result of the function: this will be the non-empty portion of the right-hand tape. This consists of two stages. First observe that the following holds:

$$\exists^! w.(w \in \mathsf{Conf} \otimes \langle x,w\rangle \in t) \vdash \exists^! r.(r \in \mathsf{W}_3' \otimes \exists^! q.\exists^! l.(q \in \mathsf{Q}_n \otimes l \in \mathsf{W}_3' \otimes \langle x,\langle q,l,r\rangle\rangle \in t))$$

Combining this with (10) yields

$$(x \in \mathsf{W}_2)^{1+\delta P}.x \in \mathsf{W}_3' \vdash \exists^! r.(r \in \mathsf{W}_3' \otimes \exists^! q.\exists^! l.(q \in \mathsf{Q}_n \otimes l \in \mathsf{W}_3' \otimes \exists^! n.(\langle n,\langle q,l,r\rangle\rangle \in d) \otimes \langle x,n\rangle \in p')) \tag{11}$$

The right-hand side of this is of the form $\exists^! r.(r \in \mathsf{W}_3' \otimes A(x,r))$

The output r is only well-formed if it consists of only 1s and 0s. The following function extracts the well-formed outputs, sending the outputs containing a blank to the empty string: and is representable in **SLST**:

$$\tau : \mathbb{N} \times \mathbb{W}_3 \to \mathbb{W}_2,\quad \begin{array}{lll} \tau(0,y) = \tau(x,\varepsilon) = \tau(\mathsf{S}x,\mathsf{S}_2y) = & \varepsilon; \\ \tau(\mathsf{S}x,\mathsf{S}_0y) & = \mathsf{S}_0\tau(x,y); \\ \tau(\mathsf{S}x,\mathsf{S}_1y) & = \mathsf{S}_1\tau(x,y); \end{array}$$

Let g be the evident term of **SLST** expressing this function as a fixpoint:

$$\langle x,y,z\rangle \in g \multimap (x = 0 \otimes z = \varepsilon) \oplus (y = \varepsilon \otimes z = \varepsilon) \oplus \exists y'(y = \mathsf{S}_2y' \otimes z = \varepsilon \oplus \exists x'.\exists y'.\exists z'.(x = \mathsf{S}x' \otimes ((y = \mathsf{S}_0y' \otimes z = \mathsf{S}_0z') \oplus (y = \mathsf{S}_1y' \otimes z = \mathsf{S}_1z')) \otimes \langle x',y',z'\rangle \in r)$$

Then

$$y \in \mathsf{N}\langle P(X^2)\rangle[p], \vdash \forall x \in \mathsf{W}'_3.\exists^! z \in \mathsf{W}'_2(\langle x,y\rangle \in g)$$

by induction over $x \in \mathsf{N}\langle P(X^2)\rangle[p]$. We leave the details to the reader, noting that the inductive step

$$\forall y \in \mathsf{W}'_3.\exists^! z \in \mathsf{W}'_2(\langle x,y,z\rangle \in g) \vdash \forall y \in \mathsf{W}'_3.\exists^! z \in W'_2(\langle \mathsf{S}x,y,z\rangle \in g)$$

uses as a lemma the fact that predecessor on words over three letters is 0-representable with multiplicity one (an easy generalization of Corollary 44).

We have

$$(x \in \mathsf{W}_2)^{1+(2.\delta P)} \vdash x \in \mathsf{N}\langle P(X^2)\rangle[p] \otimes \exists^! r.(r \in \mathsf{W}'_3 \otimes A(x,r))$$

from which

$$(x \in \mathsf{W}_2)^{1+(2.\delta P)} \vdash \exists^! y.(y \in \mathsf{W}'_2 \otimes B(x,y))$$

where $B(x,y) = \exists n.\exists^! r.(\langle x,n\rangle \in p' \otimes \langle n,r,y\rangle \in g \otimes A(x,r))$. Finally, letting f be defined as $f = \{\, z \mid \exists x \exists y.(z = \langle x,y\rangle \otimes B(x,y)) \,\}$, we have

$$(x \in \mathsf{W}_2)^{1+(2.\delta P)} \vdash \exists^! y.(y \in \mathsf{W}'_2 \otimes \langle x,y\rangle \in f)$$

where f is a term of **SLST** satisfying $\langle \mathsf{m},\mathsf{n}\rangle) \in f$ iff $\phi(m) = n$.

We have shown:

Theorem 52. *Polynomial time functions from $\mathbb{W}$ to $\mathbb{W}$ are* 0*-representable in* ***SLST*** *with domain* W *and codomain* W′ *and so are also representable.*

8. Conclusion and further work

We have a notion of provably total function in a set theory based on Lafont's Soft Linear Logic, and shown that these functions are precisely the polynomial time functions. Moreover, by using the fixpoints inherent in set theory, we have been able to give the same codomain (W′) to each represented function. One curiosity of the representation given is that input and output of total functions are given by different representations of the same set. This gives rise to an obvious question about composition. Of course, since the class of polynomial-time functions is closed under composition, so are the class of representable functions, but finding a constructive proof of this fact has proved elusive. What is known is that for any representable function f, the proof that it is representable yields a polynomial bound Q on the size of the output of f. Using a function similar to τ from the previous section, one can then extract a pre-image representation of the output of f. However, it is only in certain special cases that the proof that a function g is representable may be reworked into a proof that will take such an input—in particular, it must be 0-representable, but in addition we need to be able to give a pre-image as an argument. One special case in which this works is that of a proof via Turing representability:

Lemma 53. *Let ϕ be a function computed by some Turing machine T in polynomial time. There is a term f such that f represents ϕ and, for any term t and polynomial Q, for some polynomial P*

$$(x \in \mathsf{W}\langle Q\rangle[t])^{1+(2.\delta P)} \vdash \exists^! y.(y \in \mathsf{W}'_2 \otimes \langle x,y\rangle \in f')$$

holds, where $\langle x,y\rangle \in f'$ iff $\exists^! z.\langle x,z\rangle \in t \otimes \langle z,y\rangle \in f$.

(where $\mathsf{W}\langle Q\rangle[t]$ is the evident generalization of sets of pre-images to words).

Another evident question is the relationship between this approach to polytime and the function algebra approach; indeed, the fixpoint definitions of the tally integers and the words are named "safe" in deliberate allusion to Bellantoni and Cook's algebra BC [3]. We believe that the properties of these fixpoints more closely match those of the safe variables in BC than in the (purely logical) light logics approach [11] where a variable is safe if it is of the form $x : \S Bint$ (where *Bint* is the light logic representation of the binary integers. These variables allow a restricted form of induction, whereas our safe variables do not.

If the sets defined by fixpoint merit the label "safe", why then do the numbers over which we *do* have induction not merit the name "normal"? The answer comes from the imperfect manner in which we may encode safe recursion. Recall our first proof that multiplication is representable. In that proof, the variable $x \in \mathsf{N}$ is a side formula in the inductive step, and we obtain $!(x \in \mathsf{N})$ in result of the applied induction. In addition, of course, the number of times a variable is used is important, since we do not have unrestricted contraction.

A possible solution to the problem is to consider a more liberal notion of representation, which we call *stratified representation*.

Definition 54. A a term f is a *stratified representation* of a function $\phi : T_1 \times \cdots \times T_k \to S$ with domains $t_1, \ldots, t_k$ and codomain s if

(a) Each T_i and S are represented by t_i and s, respectively;
(b) For any $\vec{m} \in \vec{T}$ and $n \in S$ such that $\phi(\vec{m}) = n$, $\vdash \langle \vec{m}^*, n^* \rangle \in f$;
(c) There exists natural numbers $n_1, \ldots, n_k$ and $m_1 \ldots m_k$ such that

$$\vdash \forall x_1 \ldots \forall x_k . \exists^! y . (((!^{m_1}(x_1 \in t_1))^{n_1} \otimes \cdots \otimes (!^{m_n}(x_k \in t_k))^{n_k}) \multimap (y \in s \otimes \langle \vec{x}, y \rangle \in f))$$

is provable in **SLST**.

We conjecture that a stratified version of BC counting multiplicities of variables (and using a simplified variant of the cases construction in [11]) captures polynomial time. However, it has already been demonstrated in [2] that Soft Lambda Calculus with fixpoints goes beyond the representational strength of both light logics and safe recursion; it is possible, by clever choice of typing, to represent insertion sort in an intuitive fashion. It would be interesting to look at representing the operations involved as a function algebra, which by virtue of its representability in Soft Linear Logic would be immediately known to be polynomially sound.

Acknowledgments

I would like to thank Yves Lafont for some initial help, and Thomas Strahm for discussions on the work of Bellantoni and Cook.

Appendix A. Sets of preimages for addition and multiplication

The proof of Theorem 32 (that there is a set of preimages for every polynomial expression P) relied on the existence of certain sets of preimages for generalized addition and multiplication. We give here the proofs of these assumptions. In the following, $N_\theta\langle P \rangle[t]$ is the instantiation of the outermost quantifier in $N\langle P \rangle[t]$ with the term θ.

Proposition 55. *The following is provable in **SLST***:

$$x \in \mathsf{N}\langle P \rangle[t], z \in \mathsf{N}\langle Q \rangle[s] \vdash ((\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha)^{P+Q} \multimap (0 \in \alpha \multimap \exists^! u . \exists^! v . \exists ! w (w \in \alpha \otimes \langle x,u \rangle \in t \otimes \langle z,v \rangle \in s \otimes \langle u,v,w \rangle \in \mathsf{add})))$$

Proof. Given some term α of **SLST** , let

$$\beta := \{ z \mid \exists^! u . \exists^! w (w \in \alpha \otimes \langle x,u \rangle \in t \otimes \langle u,z,w \rangle \in \mathsf{add}) \}.$$

Then the following are provable in **SLST**:

(i) $\exists^! w.(w \in \alpha \otimes \langle x,w \rangle \in t) \vdash 0 \in \beta$
(ii) $\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha)^P \vdash \forall z.(z \in \beta \multimap \mathsf{S}z \in \beta)^P$
(iii) $\exists^! w.(w \in \beta \otimes \langle z,w \rangle \in s) \vdash \exists^! u . \exists^! v . \exists ! w (w \in \alpha \otimes \langle x,u \rangle \in t \otimes \langle z,v \rangle \in s \otimes \langle u,v,w \rangle \in \mathsf{add})$

These follow by elementary applications of the definition of add. The proof is completed by the derivation given in Table A1.

Proposition 56. *The following is provable in **SLST***:

$$x \in \mathsf{N}\langle P \rangle[t], z \in \mathsf{N}\langle Q \rangle[s] \vdash (((\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha))^{PQ} \multimap (0 \in \alpha \multimap \exists^! u . \exists^! v . \exists ! w (w \in \alpha \otimes \langle x,u \rangle \in t \otimes \langle z,v \rangle \in s \otimes \langle u,v,w \rangle \in \mathsf{mult})))$$

Proof. Given some term α of **SLST**, let

$$\beta := \{ z \mid (\forall x \in \alpha \exists^! u \in \alpha . (\langle x,z,u \rangle \in \mathsf{add}))^P \}$$

and let

$$\gamma := \{ z \mid \exists ! u . \exists^! w . (w \in \alpha \otimes \langle x,u \rangle \in t \otimes \langle u,z,w \rangle \in \mathsf{mult}) \}.$$

Then the following are provable in **SLST**:

(i) $\vdash 0 \in \beta$
(ii) $0 \in \alpha \vdash 0 \in \gamma$

Table A.1
Pre-image for addition

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\,(\mathrm{i})}{\exists^! w.(w \in \alpha \otimes \langle x, w\rangle \in t) \vdash 0 \in \beta} \quad \dfrac{\vdots\,(\mathrm{iii})}{\exists^! w.(w \in \beta \otimes \langle z, w\rangle \in s) \vdash A}\multimap L}{\exists^! w.(w \in \alpha \otimes \langle x, w\rangle \in t), 0 \in \beta \multimap \exists^! w.(w \in \beta \otimes \langle z, w\rangle \in s)) \vdash A}\multimap L}{0 \in \alpha, 0 \in \alpha \multimap \exists^! w.(w \in \alpha \otimes \langle x, w\rangle \in t), 0 \in \beta \multimap \exists^! w.(w \in \beta \otimes \langle z, w\rangle \in s)) \vdash A}\multimap L \quad \dfrac{}{\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha)^Q \vdash \forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha)^P}\multimap L$$

$$\dfrac{0 \in \alpha, 0 \in \alpha \multimap \exists^! w.(w \in \alpha \otimes \langle x, w\rangle \in t), \forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha)^Q, y \in \mathsf{N}_\beta\langle Q\rangle[s] \vdash A \qquad \forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha)^P \vdash \forall z.(z \in \beta \multimap \mathsf{S}z \in \beta)^P}\multimap L$$

$$\dfrac{0 \in \alpha, \forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha)^P, \forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha)^Q, x \in \mathsf{N}_\alpha\langle P\rangle[t], z \in \mathsf{N}_\beta\langle Q\rangle[s] \vdash A}\multimap R$$

$$\dfrac{x \in \mathsf{N}_\alpha\langle P\rangle[t], z \in \mathsf{N}_\beta\langle Q\rangle[s] \vdash ((\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha))^{P+Q} \multimap (0 \in \alpha \multimap A)}\forall L^2$$

$$x \in \mathsf{N}\langle P\rangle[t], z \in \mathsf{N}\langle Q\rangle[s] \vdash (\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha))^{P+Q} \multimap (0 \in \alpha \multimap A)$$

$$A := \exists^! u.\exists^! v.\exists^! w(w \in \alpha \otimes \langle x, u\rangle \in t \otimes \langle z, v\rangle \in s \otimes \langle u, v, w\rangle \in \mathsf{add})$$

Table A.2
Preimage for multiplication

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots\,(\mathrm{ii})}{0 \in \alpha \vdash 0 \in \gamma} \quad \dfrac{\vdots\,(\mathrm{iii})}{\exists^! w.(w \in \gamma \otimes \langle x, w\rangle \in t) \vdash M}\multimap L}{0 \in \alpha, 0 \in \gamma \multimap \exists^! w.(w \in \gamma \otimes \langle x, w\rangle \in t) \vdash M}\multimap L \quad \dfrac{\vdots\,(\mathrm{iv})}{\exists^! w.(w \in \beta \otimes \langle z, w\rangle \in s) \vdash (y \in \gamma \multimap \mathsf{S}y \in \gamma)^P}\multimap L}{0 \in \alpha, \exists^! w.(w \in \beta \otimes \langle z, w\rangle \in s), (y \in \gamma \multimap \mathsf{S}y \in \gamma)^P \multimap (0 \in \gamma \multimap \exists^! w.(w \in \gamma \otimes \langle x, w\rangle \in t)) \vdash M} \quad \dfrac{\vdots\,(\mathrm{i})}{\vdash 0 \in \beta}\multimap L$$

$$\dfrac{0 \in \alpha, (0 \in \beta \multimap \exists^! w.(w \in \beta \otimes \langle z, w\rangle \in s)), y \in \mathsf{N}_\gamma\langle P\rangle[t] \vdash M \qquad (\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha))^{PQ} \vdash (\forall y.(y \in \beta \multimap \mathsf{S}y \in \beta))^Q}\multimap L$$

$$\dfrac{0 \in \alpha, (\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha))^{PQ}, x \in \mathsf{N}_\gamma\langle P\rangle[t], y \in \mathsf{N}_\beta\langle Q\rangle[s] \vdash M}\multimap R^2$$

$$\dfrac{x \in \mathsf{N}_\gamma\langle P\rangle[t], z \in \mathsf{N}_\beta\langle Q\rangle[s] \vdash (\forall y(y \in \alpha \multimap \mathsf{S}y \in \alpha))^{PQ} \multimap (0 \in \alpha \multimap M)}\forall L^2$$

$$x \in \mathsf{N}\langle P\rangle[t], z \in \mathsf{N}\langle Q\rangle[s] \vdash (\forall y(y \in \alpha \multimap \mathsf{S}y \in \alpha))^{PQ} \multimap (0 \in \alpha \multimap M)$$

$$M := \exists^! u.\exists^! v.\exists^! w(w \in \alpha \otimes \langle x, u\rangle \in t \otimes \langle z, v\rangle \in s \otimes \langle u, v, w\rangle \in \mathsf{mult})$$

(iii) $\exists^! w.(w \in \gamma \otimes \langle x,w\rangle \in t) \vdash \exists^! u.\exists^! v.\exists! w(w \in \alpha \otimes \langle x,u\rangle \in t \otimes \langle z,v\rangle \in s \otimes \langle u,v,w\rangle \in \mathsf{mult})$

(iv) $\exists^! w.(w \in \beta \otimes \langle z,w\rangle \in s) \vdash (y \in \gamma \multimap \mathsf{S}y \in \gamma)^P$

(v) $(\forall y.(y \in \alpha \multimap \mathsf{S}y \in \alpha))^{PQ} \vdash (\forall y.(y \in \beta \multimap \mathsf{S}y \in \beta)^Q)$

These follow by elementary applications of the definition of add and mult. The proof is completed by the derivation given in Table A2.

References

[1] Andrea Asperti, Light affine logic, in: Logic in Computer Science, 1998, pp. 300–308.
[2] Patrick Baillot, Virgile Mogbil, Soft lambda-calculus: a language for polynomial time computation, in: 7th International Conference Foundations of Software Science and Computation Structures, part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, vol. 2987, 2004, pp. 27–41.
[3] Stephen Bellantoni, Stephen A. Cook, A new recursion-theoretic characterization of the polytime functions, Comput. Complex., 2 (1992) 97–110.
[4] A. Cantini, The undecidability of Grishin's set theory, Studia Logica 74 (2003) 345–368.
[5] J.-Y. Girard, Linear logic, Theor. Comput. Sci. 50 (1) (1987) 1–102.
[6] J.-Y. Girard, Y. Lafont, P. Taylor, Proofs and Types, Cambridge University Press, 1989.
[7] Jean-Yves Girard, Light linear logic, Inform. Comput. 143 (2) (1998) 175–204.
[8] Jean-Yves Girard, Andre Scedrov, Philip J. Scott, Bounded linear logic: a modular approach to polynomial-time computability, Theor. Comput. Sci. 97 (1) (1992) 1–66.
[9] V.N. Grishin, A nonstandard logic and its application to set theory, Studies in Formalized Languages and Nonclassical logics, Nauka, 1974, pp. 135–171 (in Russian).
[10] V. N Grishin, Predicate and set-theoretic calculi based on logic without contractions, Math. USSR Izvetija 18 (1981) 41–59.
[11] A.S. Murawski, C.-H.L. Ong, On the interpretation of safe recursion in light logic, Theor. Comput. Sci. 318 (2004) 197–223.
[12] M. Shirahata, Fixpoint theorem in linear set theory, 1999.
[13] K. Terui, Light affine lambda calculus and polytime strong normalization, in: Proceedings of the 16th Annual IEEE Conference on Logic in Computer Science, 2001, pp. 209–220.
[14] Kazushige Terui, Light affine set theory: a naive set theory of polynomial time, Studia Logica 77 (1) (2004) 9–40.
[15] Yves Lafont, Soft linear logic and polynomial time, Theor. Comput. Sci. 318 (1–2) (2004) 163–180.